RENDERING AND DEPLOYMENT WITH REACT

FULL STACK SKILLS BOOTCAMP

APIS, CONDITIONAL RENDERING, LAYOUTS, AND DEPLOYMENT

- **Lesson Overview:**
- In this lesson, we will be introduced to:

1. Using 3rd party APIs with React
2. Conditional Rendering
3. App Layouts
4. Deployment with render.com

USING 3RD PARTY APIS IN REACT WITH FETCH

- **What is fetch?**
  A JavaScript function for making HTTP requests to APIs.
- useEffect used to fetch data after the component mounts.
- Data is stored in state and displayed in the component.

  Demo…

```
import { useState, useEffect } from 'react';

function App() {
  const [data, setData] = useState(null);

  useEffect(() => {
    fetch('https://api.example.com/data')
      .then(response => response.json())
      .then(data => setData(data))
      .catch(error => console.error('Error fetching data:', error));
  }, []);

  return <div>{data ? JSON.stringify(data) : 'Loading...'}</div>;
}
```

CONDITIONAL RENDERING IN REACT

- **What is Conditional Rendering?**
  A technique to render elements or components based on certain conditions.
- **Common Patterns**:

Ternary Operator:

```
{isLoggedIn ? <Dashboard /> : <Login />}
```

Logical AND (&&):

```
{isAdmin && <AdminPanel />}
```

CONDITIONAL RENDERING IN REACT

- Render loading spinner until data is loaded:

  demo…

```
function App() {
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetchData().then(() => setLoading(false));
  }, []);

  return <div>{loading ? <Spinner /> : <Content />}</div>;
}
```

APP LAYOUTS USING PROPS.CHILDREN

- **What is props.children?**
  A special prop in React that allows components to render their children elements.
- **Creating a Layout Component**:
  Define a reusable layout that wraps content passed as children.
- **Use Cases**:
  - Wrapping multiple pages with a consistent header/footer structure.
  - Creating reusable components that serve as "wrappers" for various sections of the app.

  Demo…

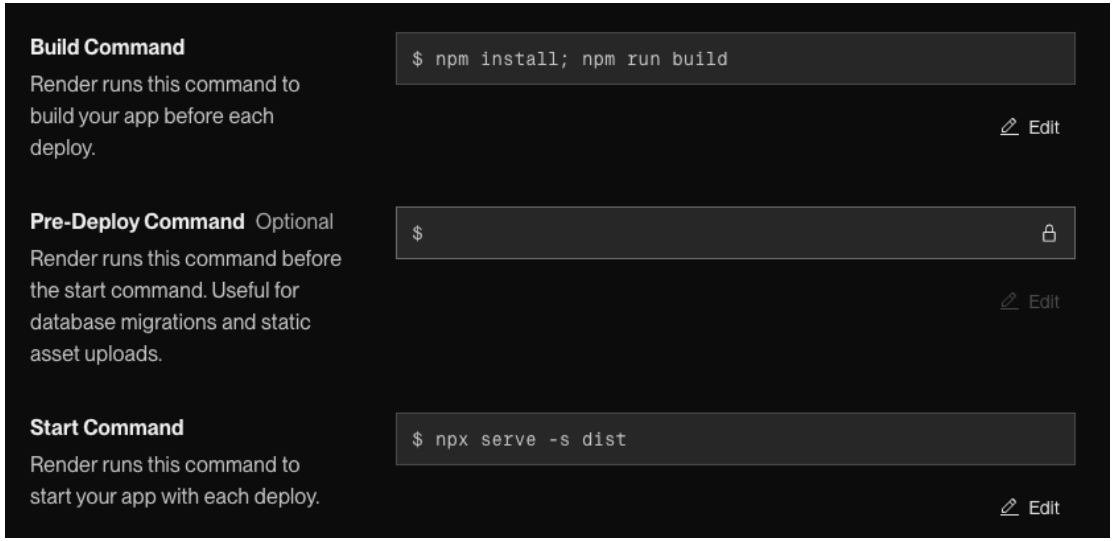
```
function Layout({ children }) {
  return (
    <div className="layout">
      <Header />
      <main>{children}</main>
      <Footer />
    </div>
  );
}

function App() {
  return (
    <Layout>
      <HomePage />
    </Layout>
  );
}
```

DEPLOYMENT USING RENDER.COM

- **Why render.com?**
  Render.com is a platform that simplifies deploying web applications, including React apps.
- Provides automatic deployment, scaling, and HTTPS out of the box.

DEPLOYMENT USING RENDER.COM

- **Steps to Deploy a React App**:**Step 1**:
  Set up your React app and push it to a Git repository (GitHub or GitLab).
- **Step 2**:
  Sign up on Render.com and create a new web service.
- **Step 3**:
  Connect your repository and configure the build settings:
  - **Build Command**: npm install && npm run build
  - **Start Command**: serve -s build (assuming you're using serve to serve the static files)
- **Step 4**:
  Deploy and monitor the logs to confirm successful deployment.



CONCLUSION

- **Using APIs**:Fetch and display data from third-party APIs with fetch and useEffect.
- **Conditional Rendering**:Techniques to render components based on conditions, enhancing user experience.
- **Layouts with props.children**:Efficiently structure apps by passing child components into layout wrappers.
- **Deployment**:Render.com simplifies deploying and managing React apps in production.

Step8Up Academy

QUESTIONS?